Chapter One: Contents

(Network – 10 December 2002 – LA-UR 01-5712 – Portland Study Reports)

Chapter One—Network

1. INTRODUCTION

The TRANSIMS network tables were created from data supplied by Portland Metro. The data, provided to LANL in ESRI Shapefiles, were reprojected from the Oregon State Plane coordinate system (Zone 5076) with measurements in feet to Universal Transverse Mercator (UTM) coordinates (Zone 10) with SI units. Both projections utilized the NAD83 datum. ESRI's ARC/Info 7.1.1 and ArcView 3.1 were extensively utilized to manage the geographic data. After the tables were completed, they were checked with the *NetworkValidator* utility (see the general documentation for more information). The tables utilized in the case study ranged in size from 19 kilobytes (KB) to 35.6 megabytes (MB), depending on the numbers of records and attributes each table contained. Table 1 provides a list of table sizes and number of records.

Table 1. Size and number of records contained in each TRANSIMS network table.

Table	Number of Records	Size (MB)
Activity Location	243,423	35.6
Detector	10,830	0.5
Lane Connectivity	507,700	12.7
Link	124,904	16.8
Node	100,539	3.5
Parking	121,504	8.5
Phasing Plan	33,871	1.1
Pocket Lane	4,379	0.16
Process Link	526,094	24.4
Signal Coordinator	2,086	0.02
Signalized Node	2,081	0.08
Timing Plan	1,716	0.05
Transit Stop	9,827	0.56
Unsignalized Node	237,343	4.7

2. LINK TABLE

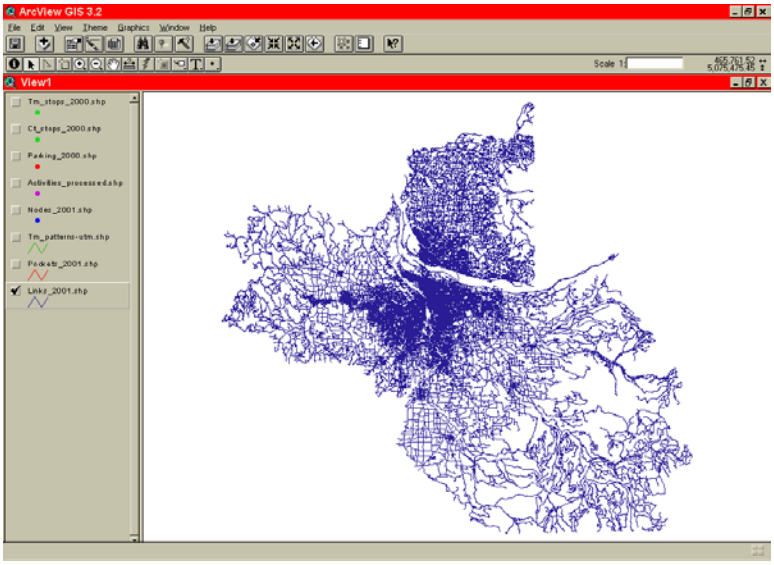

Fig. 1. Extent of the Portland link network.

2.1 Data

The link data set, as seen in Fig. 1, was based on an enhanced version of the U.S. Census Bureau's Topologically Integrated Geographic Encoding and Referencing (TIGER) road network. These street data were utilized because Metro's Data Resource Center (DRC) maintains a continually updated version for regional planning purposes. The TIGER data contained the following link attributes needed for the TRANSIMS table:

- `ID`
- `NAME` – The TIGER attributes street prefix, name, type, and suffix, which were concatenated in ArcView to form the street name.

Several of the link attributes were transferred to the street network from Metro's EMME/2 modeling network, which is maintained for travel forecasting applications. These included:

- `PERMLANESA/B` – Links designated as walkways did not have any permanent lines in either direction.
- `CAPACITYA/B`
- `SPEEDLMTA/B` – The actual speed limit of a street segment was used, with the exception of several specific instances. Because of the way the speed limit is interpreted by the microsimulation, some speed limits were artificially raised in order to have more realistic traffic behavior. Freeways and some ramps were assigned the

speed limit of 36 meters per second (m/s) (80 mph). Links that allowed speeds less than 11 m/s (25 mph) were increased to this threshold. Table 2 shows the minimum speeds used by each function class. Not included are 115 links that were assigned low speeds (4 or 5 m/s) to discourage travel on these streets in order to obtain more realistic travel patterns.

Table 2. Function class minimum speeds.

Function Class	Number of Links	Minimum Speed (m/s)
COLLECTOR	3231	11
FREEWAY	1604	36
LIGHTRAIL	58	11
LOCAL	104693	11
PRIARTER	5789	11
RAMP	1552	11
SECARTER	6839	11
WALKWAY	129	0
XPRESSWAY	894	11

- FUNCTCLASS
- VEHICLE – Autos, buses, and trucks were allowed on every link that was not specified as a bus-only or light-rail-only link in the EMME/2 network. Each link mode was further cross-checked with the transit routes to assure that the specified transit vehicle was permitted on the required links. Refer to Section 7 (*Transit Tables*) for more information. Pedestrians were allowed on all streets, except freeways, expressways, and ramps.

Any link without an equivalent in the EMME/2 network was assumed to be a local street. These links were assigned one lane in each direction with a speed limit of 25 mph (11 m/ps), a capacity of 500 vehicles per lane, and the function class LOCAL.

The remaining attributes were manually added or generated by Metro or LANL:

- NODEA/B
- LEFTPCTSA/B and RGHTPCKTSA/B
- TWOWAYTURN
- LENGTH – Calculated by ArcView and/or ARC/Info.
- GRADE – Based on the elevations of the link end nodes (Node A and Node B). It was calculated by taking the difference in elevations from Node B to Node A and dividing it by the link length. Refer to Section 3 (*Node Table*) for more information on node elevations.
- SETBACKA/B – Set to zero for the case study.

- `FREESPDA/B` – On most links, the vehicles were allowed to travel at speeds up to 1.25 times the link's speed limit.
- `THRUA/B` – Default "through" link was determined by the *PrepareNetwork* program and automatically added to the link table.
- `COLOR` – No longer utilized by the microsimulation so was given the default value of zero.
- `NOTES` – Given the default entry `NONE`.

The completed link table, which contains approximately 124,900 records, is 16.8 MB.

2.2 Problems

The link network generally contained two error types: human mistakes and inaccuracies built into the data based on its source. Errors caused by human interaction included:

- Reversed lane connectivity
- Inconsistent link attributes, such as the speed limit and number of lanes, for continuous sections of roadways encompassing multiple links

The main source of error is due to the fact that the network is based on the TIGER street data. The TIGER data, which historically have been digitized from the U.S. Geological Survey (USGS) topographic maps for the Census Bureau, contain many topological errors and are currently not designed for this type of application.

- Several ramps, cul-de-sacs, and other road segments began and ended at the same node. Since the microsimulation acquires a link's geographic location from its end nodes' coordinates, a link that begins and ends at the same place appears to be only a point. Many of the local street cul-de-sacs were eliminated since this would not greatly alter the microsimulation results. More important features, such as ramps, were divided into multiple links by inserting new nodes as shown in Fig. 2

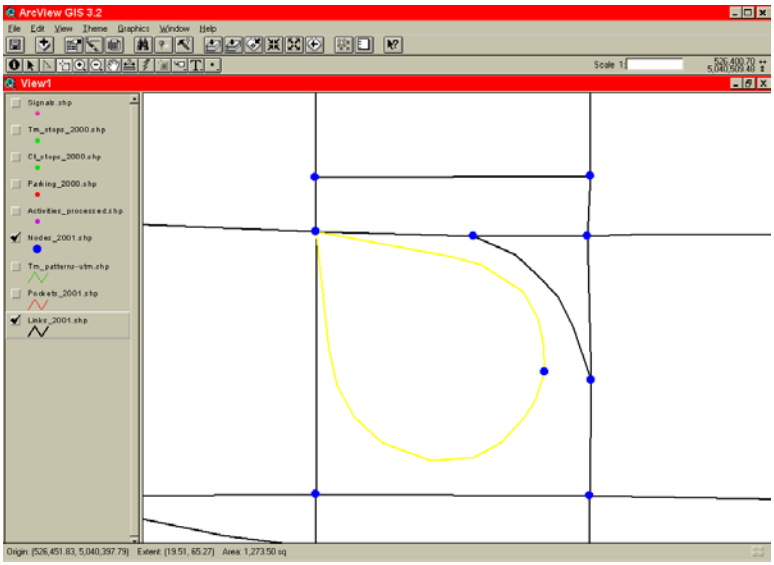

Fig. 2. Ramp that was both divided into two pieces and had a node split into two to separate the overpass and the underpass links.

- Because the TIGER network is only a two-dimensional data set, overpasses and underpasses are connected together by nodes, which were inserted by ARC/Info whenever two road segments crossed. Also, a ramp (Fig. 2), comprised of multiple, continuous links, is sometimes connected to both an underpass and an overpass by the same node. If not corrected, simulated vehicles could legally drive off an overpass onto an underpass and avoid any ramps. To resolve these situations, the node in question was divided into two or more nodes as needed; each node exists at a different elevation. Refer to Section 3 (*Node Table*) for more information.
- Multiple links that have matching beginning and ending nodes, such as a side street shown in Fig. 3 that branches off a main road and then rejoins it, appear in the microsimulation to be duplicated copies of the same link. New nodes were inserted into the secondary links to divide them into new links with different end nodes.

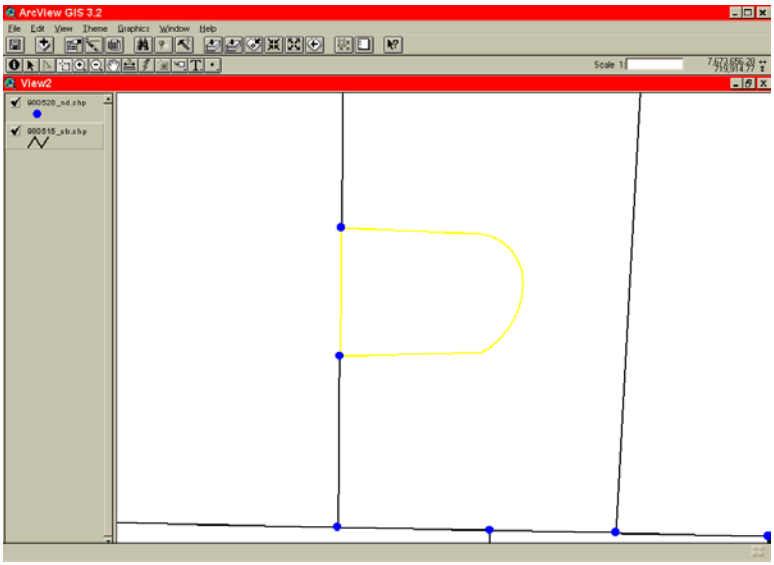

Fig. 3. Two links sharing the same end nodes.

- At some nodes, links appeared to be connected to one another but were in fact not attached. Instead, each of the links ended at a different node; the nodes were located in close proximity to each other. The links, such as those in Fig. 4, had to be manually joined together, or simulated vehicles could not travel across the disjointed links.

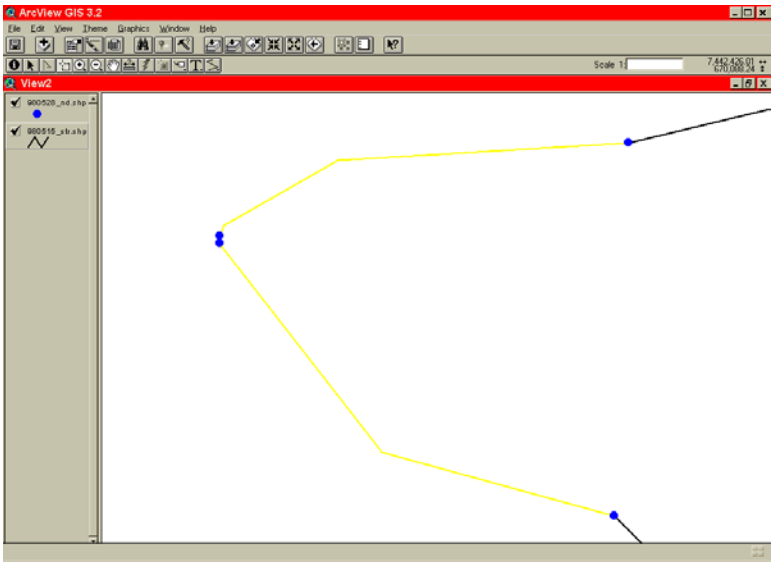

Fig. 4. Two links' disjointed links that should be connected.

- In a few instances, street segments were composed of overlapping links. Instead of having a series of links that connected end to end, sections of the links were repeated by other links. Since this could cause serious problems when routing trips, the overlapping links were removed.

- Links with a very short length caused various traffic problems. In some cases, vehicles did not have enough room to move from one lane to another when needed, such as when a vehicle has to move into a turn lane. This resulted in traffic jams from the vehicle becoming stationary or the vehicle becoming lost. Because of editing time restraints, all links less than five meters were removed from the network by merging them with longer links. Links that ranged in length from five meters to less than 15 meters were assigned a link length of fifteen meters. The topology of these links was unchanged. A further length modification involved extending the freeway and expressway links. The length of any freeway or expressway link that had a length less than the sum of the link's length and speed limit was increased to that value.

- Intersections that were comprised of multiple links converging at multiple nodes needed to be merged into one node. For example, a divided expressway, as seen in Fig. 5, is often represented with two one-way links. At areas where another road intersects the expressway, a small link is needed to cross from one expressway section to the other, thereby requiring the intersection to be comprised of multiple nodes. For traffic signals and signs to operate properly, these multiple node intersections must be condensed down to join at only one node.

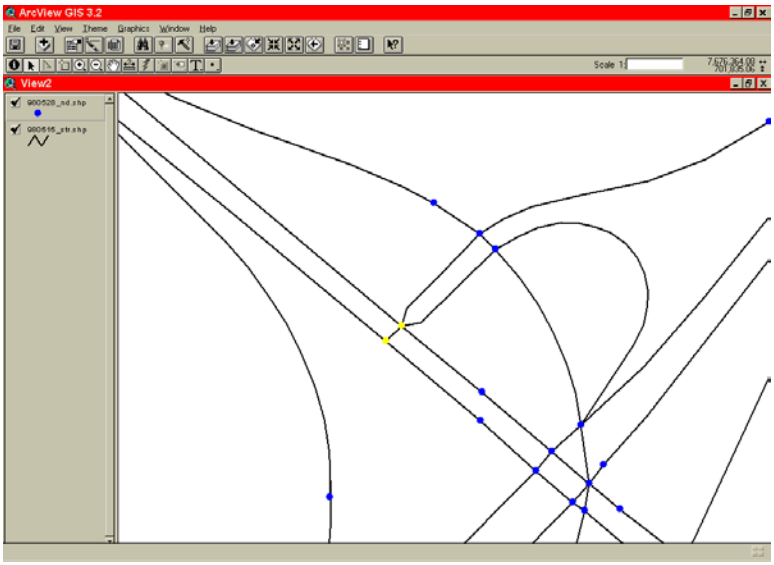

Fig. 5. An intersection with a signal that is spread over multiple nodes.

- Alleys, contained in the TIGER data, were eventually removed from the network since the links were short and would have had a minimal impact on traffic flows.

3. NODE TABLE

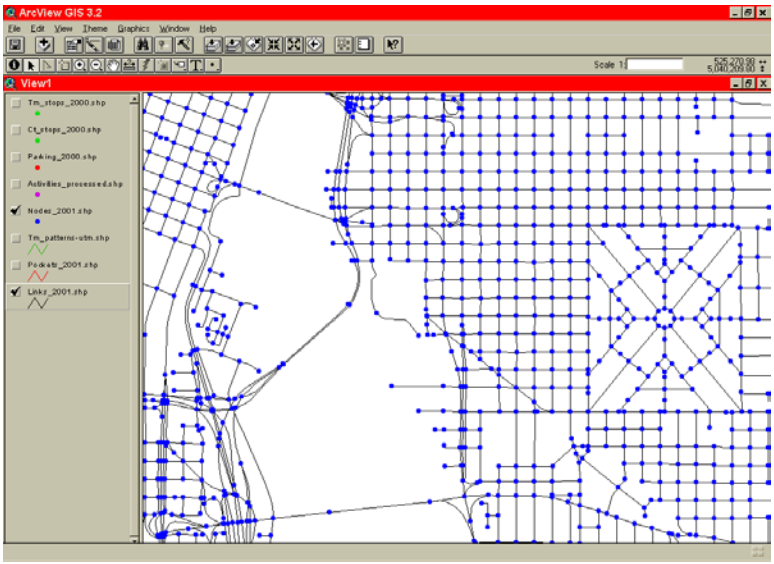

Fig. 6. Nodes in the Portland network.

3.1 Data

Along with the link data, the node data, shown in Fig. 6, are also part of the TIGER road data set. The two-dimensional data set was supplemented by the addition of the elevations for each node. The TIGER node data contained the required TRANIMS data.

- `ID`
- `EASTING` – Inserted using the ArcView commands [Shape].GetX in the Calculate function.
- `NORTHING` – Inserted using the ArcView command [Shape].GetY in the Calculate function.
- `ELEVATION` – Initial node elevations' values were determined from the region's soil surface elevation grid. Surface road elevations were assumed to be approximately equal to the soil surface elevations. The elevations for below- and above-grade structures, including ramps, overpasses, bridges, and tunnels, were estimated based on the soil surface elevations. At locations where links cross but in reality do not actually intersect (i.e., overpasses and underpasses), each node was split into multiple nodes existing at different elevations. Aerial photographs and fields checks were utilized to confirm which links crossed above or below other links that these intersections. The node elevations of these links were also checked to verify that the values provided sufficient vehicle clearance where appropriate. Bridge elevations for the Willamette and Columbia Rivers were calculated from available maximum height values that were added to the water surface elevations.

- `NOTES` – Given the default entry `NONE`.

The completed node table, approximately 3.5 MB in size, contains nearly 100,500 records.

3.2 Problems

As with the TIGER link network, the node problems were related mainly to conflicts between the TIGER and TRANSIMS formats. In areas where overpasses and underpasses cross, the two-dimensional TIGER network places a node, forcing the links to intersect. During the initial splitting process, several nodes, which needed to be separated, were missed. This resulted in the nodes having to be divided at a later date during testing, which in turn affected the microsimulation results.

4. Pocket Lane Table

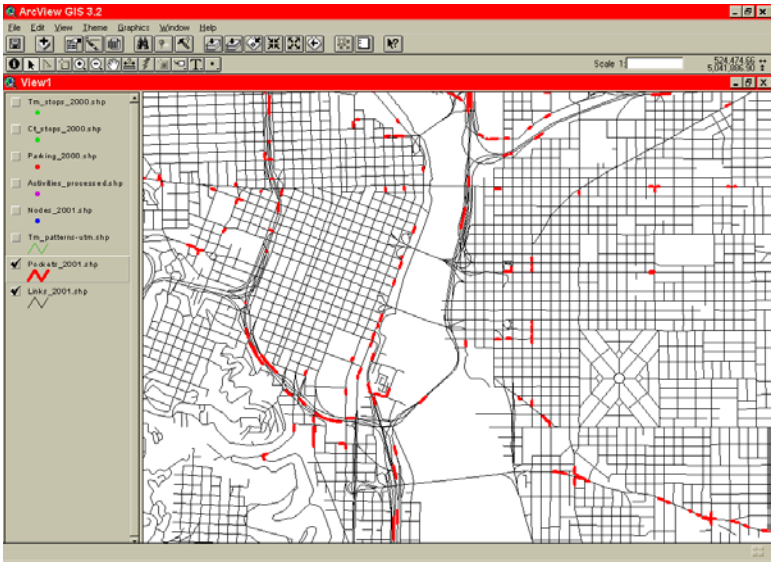

Fig. 7. Pocket lanes in the Portland network.

4.1 Data

The pocket lanes (Fig. 7) were created by Metro based on a series of four-foot resolution aerial photographs covering the entire Portland metropolitan area. Metro created a tool in ArcView that allowed the user to draw and capture the pocket lanes as they appeared in the aerial photos. The tool also prompted the user to input attribute information—which node the pocket lane led towards, side of street the pocket is located, type of pocket lane, etc. This information was output into an ArcView shapefile, which in turn was transformed into the pocket lane table. Data output by Metro's tool for direct use in the pocket lane table included:

- `ID`
- `NODE`
- `LINK`
- `OFFSET` – Measured using an ArcView tool.
- `STYLE` – All pocket lane styles were used.
- `LENGTH` – Initially calculated by ArcView.

Other fields added later were:

- LANE – The lane number was determined from a combination of the number of permanent lanes and pocket lanes on one side of a link. The number of right and left pocket lanes for each type on a side of a link was summed. Since one turn lane, merge lane, and pull-out can each have the same lane number, assuming they do not overlap, the largest sum value for each of the right and left pockets was saved in the links table as the number of pocket lanes in a particular direction (LEFTPCKTA, RGHTPCKTB, etc.). The lane numbers in the pocket lane table are based on the number of lanes and the permanent lanes. Because lanes are numbered from the middle to the outside of a link, left turns are numbered first (i.e., lane 1, lane 2, etc.). Permanent lanes are numbered next (lane 3, lane 4, etc.), and right pocket lanes are numbered last (lane 4, lane 5, etc).
- NOTES – Given the default entry NONE.

The completed pocket lane table, which contained approximately 4400 records, was 158 KB.

4.2 Problems

Because of the way the TIGER road network fragmented the streets into link segments, several problems emerged while creating the pocket lanes.

- Pocket lanes sometimes, in reality, crossed several links. Because TRANSIMS pocket lanes can exist only as a portion of one link, the pocket lane had to be redrawn to accommodate the network topology. Further time for link topology improvements would have resulted in more accurate pocket lane lengths.
- Several features, such as large parking lots, often utilize turn pockets for entering and exiting. However, if no street leading to the parking lot exists in the street network, these turn pockets should not be included in pocket lane data set. Turn pockets of this type that were included in the pocket lane table were removed or changed to pullout lanes.
- Intersections that contained multiple offset links, such as divided highways, sometimes had the pocket lane assigned to the wrong link. For example, the small link between the two divided highway links was forgotten, so the turn/merge lane was placed in the wrong location.

Other problems included:

- The incorrect NODE was listed in the table, so the microsimulation placed the pocket lane at the wrong end of the link.
- Pocket lanes were not updated when the link network data set was corrected. This sometimes resulted in pocket lanes no longer being connected to intersections with turns because a link was inserted between the pocket lanes and the intersection.

- Merge lanes were added to all freeways connected to ramp links in order to obtain more realistic traffic patterns.

5. ACTIVITY LOCATION TABLE

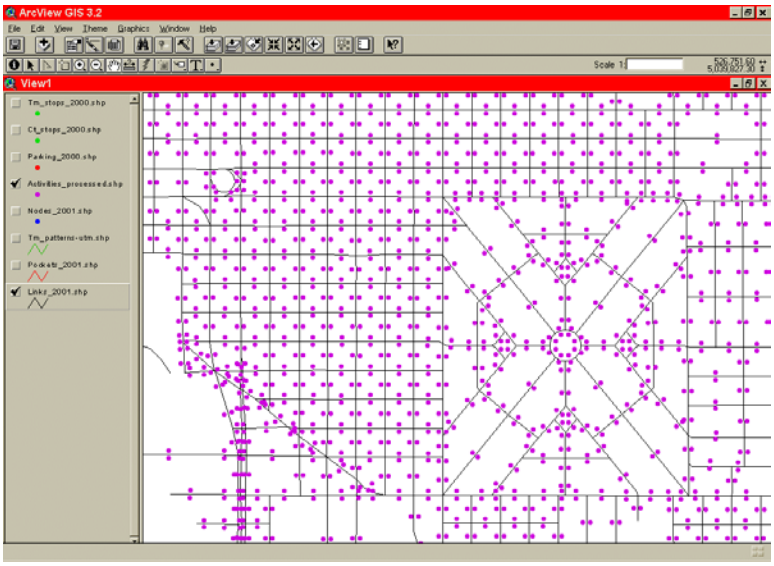

Fig. 8. Activity locations in the Portland network.

The activity locations, shown in Fig. 8, utilized in the case study are generic points representing housing and work sites. Two activities are placed on each link; each activity is located ten meters to one side of the link mid-point. Generally, all links have activity locations except bridges, ramps, and freeways. Extra boundary activities, used for routing, were placed in specific traffic zones and on network edges where no activity locations existed, such as on a freeway.

Each activity location has two types of data associated with it: required and optional. The mandatory data set includes information that provides the position of the activity locations.

- `ID`
- `NODE`
- `LINK`
- `OFFSET` – Placed in the center of the link so the offset was half the link length.
- `LAYER` – The `WALK` layer was utilized as the default mode.
- `EASTING` – Inserted using the ArcView commands [Shape].GetX in the Calculate function.
- `NORTHING` – Inserted using the ArcView command [Shape].GetY in the Calculate function.
- `NOTES` – Given the default entry `NONE`.

The accessory data, utilized to route travelers, are optional. The type of data included, if any, is up to the user to define. For the Study, one set of activity data was created from various types of geographic data. This data set was then processed to create the attractors employed when routing travelers throughout the network. The original unprocessed activity location attributes are:

- Traffic Analysis Zones (TAZ) – Provided by Metro. The TAZ values were added to the activity file using *identity*, a point-in-polygon command in Arc/INFO. The activity location is the point, and the TAZ data are the polygons.
- Census Tract and Block Group Numbers – Added using the same procedure as in the inclusion of the TAZ values.
- Households – The number of households on a link was provided by Metro employing the following procedure:

  1) Create a database of single- and multi-family parcels from digital tax lot data.
  2) Calculate the percentage of total parcel area that one parcel occupies inside a TAZ.
  3) Allocate households to each parcel based on parcel size by multiplying the total number of households in a TAZ by the parcel percentage calculated in Step 2.
  4) Distribute parcels to the street network. The links, excluding freeways and ramps, were segmented into smaller units based on each link vertex. Parcel centroids were matched to the nearest segment midpoint. The link segmentation improved the occurrence of the parcels attaching to the appropriate link.
  5) Sum the number of households on a link from the link segments.
  6) Since there are two activities per link, the number of households on a link was equally divided between both activities.

- Schools – Provided by Metro. Each public and private elementary and second school was assigned to the nearest activity location. The activity location table shows what type of school is assigned to each activity: 0 = none, 1 = elementary, 2 = middle/junior high school, and 3 = high school.
- College and University Enrollments – Allocated to the activity location nearest to the schools' geographic positions. Some schools were spread over multiple campuses so each area was given a percentage of the total enrollment of the school.
- Number of Employees – To determine attractors for work locations, Metro provided the locations of companies operating in the case study area. Each company was assigned to the nearest activity location. The businesses were characterized by their Standard Industry Classification (SIC) code. Then, the total employees for each SIC code category (Service, Retail, Government, College, and Other) at an activity location was summed. These SIC category counts, as well as the total number of workers employed at that location, were included in the activity location table.

- Park Area – The total area surrounding an activity location that is designated park or recreation land was supplied by Metro.
- Number of Households Per Acre – Calculated by first summing the total number of households in each TAZ. Each sum was divided by its respective TAZ's total area (in acres). Finally, the activities were assigned a value based on the TAZ in which it was located.
- Distance to Nearest Transit Stop – The Euclidean distance between an activity (X_A, Y_A) and the nearest transit stop (X_T, Y_T) was calculated using the following formula:

$$Dist = \sqrt{(X_A - X_T)^2 + (Y_A - Y_T)^2}$$

- Trip Origin/Trip Destination – Because some of the boundary activity locations have been placed on the edge of the network on one-way links, such as freeways, routing problems resulted in some travelers not being able to complete their trips. These Boolean columns were utilized to show whether each activity could be used as a trip origin or destination location to avoid this problem.
- River Zone – Arbitrary zone number assigned for analysis purposes.
- Urban Type – Arbitrary zone number assigned for analysis purposes.
- Parking Zone – Arbitrary zone number assigned for analysis purposes.

The initial activity location attributes were processed with an algorithm to generate a new set of attractors. The number of households, locations of schools, college and university enrollment, and the number of employee types were replaced with: Home, Shop, Visit, Social, Other, Serve Passenger, and College. The land use and employment variables used to create these attractors are described in Volume Two (*Study Setup: Parameters and Input Data*), Chapter Three (*Activity Generator*) of the Study.

The other original attributes were included in the final activity location table without being altered. The final table, which contained approximately 243,400 records, is 35.6 MB.

6. PARKING

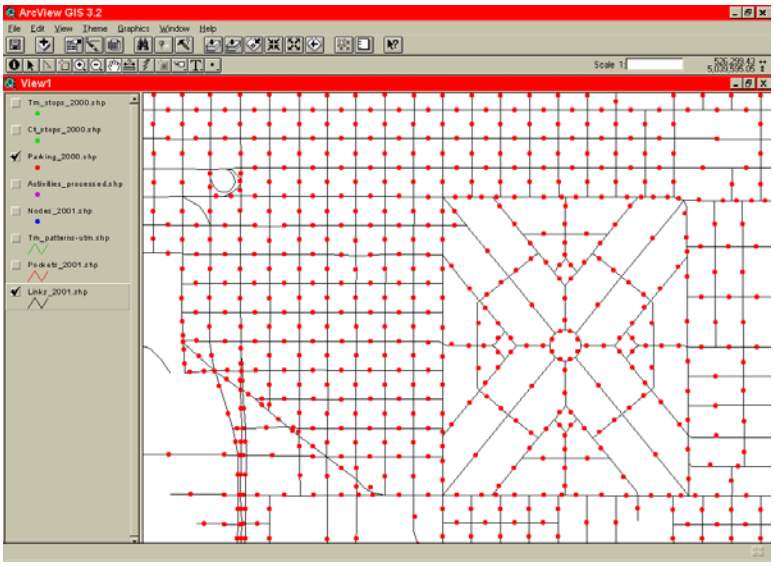

Fig. 9. Generic parking locations in the Portland network.

Parking locations, lots, and network boundary places, were created for the microsimulation. Generalized parking lots (Fig. 9) were placed on links not designated as ramps, bridges, freeways, or walkways. Special parking lots were created for some bus-only and light-rail-only links so transit vehicles would have a place to begin and end transit routes. Network boundary parking locations were added to specific boundary zone areas needed for modeling purposes and at network edges where parking did not exist, such as on freeways. Finally, generic park and ride lots were added to links with actual park and ride lots, the locations of which were provided by Metro. The parking table contains the following data:

- `ID`
- `NODE`
- `LINK`
- `OFFSET` – Placed in the center of the link so the offset was half the link length.
- `STYLE` – Lots, park and ride, and network boundary parking were used.
- `CAPACITY` – All lots were assigned unlimited parking capacities.
- `GENERIC` – Used in order to simplify the microsimulation.
- `VEHICLE` – Modes allowed were autos, buses, light rail transit, and trucks.

- `STARTTIME/ENDTIME` – This feature is not currently being utilized by microsimulation, so there are no enforced parking restrictions.
- `NOTES` – Given the default entry `NONE`, except for boundary lots and those lots used only by transit vehicles.

The parking table, which contains approximately 121,500 lots, is 8.5 MB.

7. TRANSIT TABLES

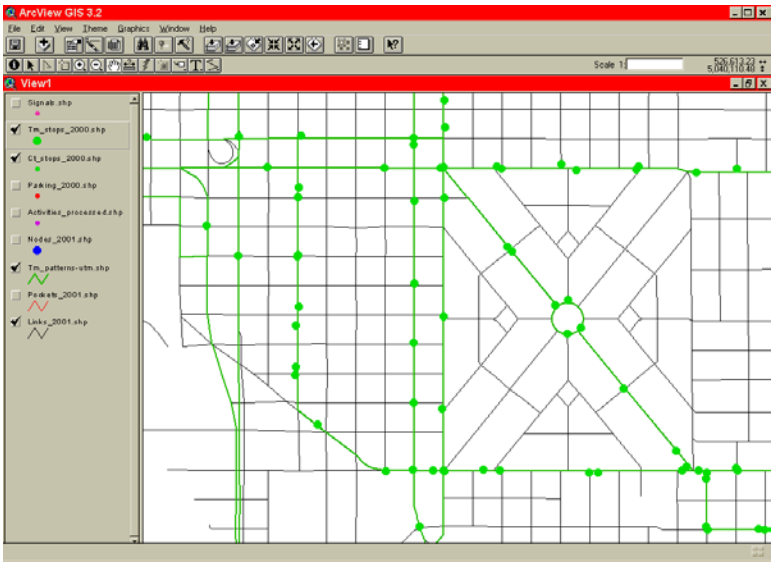

Fig. 10. Transit stops and routes in the Portland network.

Metro collected the transit data from two agencies within the Portland metropolitan area: Tri-Met, which primarily serves the Oregon counties, and C-Tran, which provides service mainly to Clark County in Washington.

7.1 Transit Stop Table

7.1.1 Data

The transit stops for most of Portland were provided to Metro by Tri-Met in ArcView shapefiles, shown in Fig. 10. Initially, the Tri-Met stops contained multiple records for the same stop if it was utilized by more than one transit route. A custom program, written by Metro, created the transit routes (see Section 7.2 (*Schedule File*)) and output the transit stop information, with one record per stop, in the proper TRANSIMS format. The transit stop data provided to Metro by C-Tran was in text format. Transit stop locations were manually geocoded onto the link network. Metro provided a condensed version of all transit stops, containing only one record per stop as required by TRANIMS formatting.

The transit stop table was created with the following fields:

- `ID`
- `NAME`
- `NODE`

- `LINK`
- `OFFSET` –The custom routing tool that created the master transit stop list also calculated the link, node, and offset information and saved the information.
- `VEHICLE` – Only buses and light rail transit were modeled.
- `STYLE` – Only stops used; no stations were included.
- `CAPACITY` - Set to zero for an unlimited number of vehicles allowed at a transit stop.
- `NOTES` – Given the default entry `NONE`.

The transit stop table, which contains approximately 9800 stops, is 563 KB.

7.1.2 Problems

Because of errors in the link network, several of the transit stops were coded on the wrong side of the street. This caused problems later on when trying to build the Transit Route file and the Transit Driver plans. To correct the problem, the stop was moved to the other side of the street by changing the `to` node in the transit stop table.

7.2 Schedule File

7.2.1 Data

The schedule information for the Oregon buses and light rail vehicles was provided to Metro by Tri-Met. The file contained the time each transit vehicle stopped at every transit stop along all routes for an entire day on a typical weekday. C-Tran supplied Metro with a copy of their printed bus schedule and a list of all stops in a bus route, including those stops not listed on the schedule. Metro provided LANL with the complete Tri-Met itinerary and an electronic version of the C-Tran printed schedule, as well as the list of stops for each route.

To complete the C-Tran itinerary, a custom C++ program was written that calculated the stop times for all stops in each route based on the printed time table, the complete list of stops in each route, and links composing each bus route. The time to travel between each stop in the route was calculated based on the length and speed limit of each link in the route. This time was then added to the known stop times in order to estimate the time of arrival at the other stops. A final itinerary for the C-Tran buses was output from the program. The itineraries for the Tri-Met and C-Tran transit vehicles were combined and sorted (in the order shown below) to create the transit schedule file.

- Route ID
- Time
- Stop ID

The final table, which contained approximately 495,000 records, was 7.4 MB.

7.2.2 Problems

When the links and nodes composing each transit route were compared with the order of stops visited in each route, several of the stops were discovered to be located outside of the route path or placed on the wrong side of the street. The stops were either moved onto the path of the transit route by moving it to a new link and/or node, or the stop was removed from the transit schedule.

7.3 Transit Route Table

7.3.1 Data

The transit routes for the Tri-Met and C-Tran data, as seen in Fig. 10, were created by a custom ArcView Network Analyst tool written by Metro. The tool determined the shortest path between transit stops to construct the entire transit path on the link network. In the instances when the shortest path algorithm did not properly replicate the transit route, dummy stops could be added manually to force the path to use the correct network links. Metro provided the transit route data in shapefiles and database tables so the route could be represented by a list of links and `to` nodes in order. A custom C++ program was written to check the routes to ensure that the connectivity was correct. This included verifying that the links and nodes were connected to each other in the proper order, and that each link had at least one lane for the transit vehicle to drive on. The program also output the transit route table in the proper format.

- Route ID – Assigned by Metro when assembling routes.
- Number of Stops in Route – Summed by C++ program.
- Transit Type – Bus or Light Rail.
- Stop ID
- Link ID
- Node ID
- Transit Zones –Determined from the Tri-Met and C-Tran transit fare zones. A separate table, the Transit Zone table, contains the costs of traveling between each zone.

The table, containing 626 routes, is 623 KB in size.

7.3.2 Problems

Because of link network repairs, the transit routes sometimes had connectivity problems. The connectivity of the route links and nodes were checked frequently to ensure that the route paths were correct.

7.4 Transit Driver Plans

7.4.1 Data

The transit driver plans were assembled by a custom C++ program. The program combines the transit route node paths, transit schedule starting and ending times, parking data for route starting and ending locations, and the link data to include path information to and from the starting and ending parking lots into the format required by the microsimulation.

- Traveler ID – The traveler IDs for the transit drivers were sequentially numbered, beginning with ten million. This number was chosen so that the transit drivers would not have the same traveler IDs as the synthetic population.
- User Field – Given the default value 0 since this is not used by the microsimulation.
- Trip ID – Assigned the default value 1 since each transit driver plan contained only one trip.
- Leg ID – Given the default value 1 since each trip only had one leg.
- Activation Time – Calculated by subtracting 45 seconds from the time the transit vehicle leaves the first stop.
- Start Accessory ID – All transit routes began at a parking lot. The custom C++ program located the parking lot nearest to the first transit stop and assigned the parking lot to be the starting accessory.
- Start Accessory Type – All start accessories were parking lots, so the default value 2, which represents parking lots, was used.
- End Accessory ID – All transit routes end at a parking lot. The custom C++ program located the parking lot nearest to the last transit stop and assigned the parking lot to be the end accessory.
- End Accessory Type – All ending accessories were parking lots, so the default value 2, which represents parking lots, was used.
- Duration – Determined by subtracting the Activation Time from the Stop Time.
- Stop Time – Calculated by adding 45 seconds to the time the transit vehicle arrives at the last stop in the route.

- Max Time Flag – Boolean set to `TRUE`.
- Cost – Given the default value of zero.
- CGF – Same as the Duration, so that value was inserted.
- Driver Flag – Boolean set to `TRUE` since the plan is for the transit driver.
- Mode – Set to 1, since this is the value used for the transit mode of travel.
- Number of Tokens – The number of values in the mode-dependent data for a transit driver, including a count for the token itself, the Schedule Pairs digit, Vehicle ID, Route ID, number of nodes in the route, and the number of Stop IDs and Departure Times in the Schedule Pairs.
- Schedule Pairs – No schedule information was used to route the transit vehicles, so this was set to zero.
- Vehicle ID – Set to the same number as the Traveler ID in order to avoid conflicts with the auto vehicle ids.
- Route ID
- List of Node IDs – The nodes, in order, were gotten from the route data. If the beginning and ending parking lots were not on the same node as the beginning and ending transit stops, those nodes were added to the list of nodes.
- List of Schedule Pairs – Not used in the microsimulation.

The Transit Driver Plan file, which contains approximately 9400 plans, is 9.1 MB.

7.4.2 Problems

Because the transit driver plans rely on data from so many data sources, any changes to the links or nodes in the transit path, transit schedule, or the parking could result in the transit driver plans being corrupted. With so many variables, it was sometimes difficult to track down problems. Also, if the beginning parking lot is located on the same link as the first transit stop, the first node in the transit route must be located past the parking lot in the route. When executing each transit driver plan, the vehicle leaves the starting parking location and heads towards the first node in the route. If the first node is before the parking lot, the vehicle will have to make a U-turn to continue onto the first transit stop. A similar problem occurs if the final node in the route is located after the ending parking lot in the route.

7.5 Transit Vehicle File

The vehicle plan file was created by a custom C++ program that input the transit driver plan file and output the vehicle plan file in the proper format.

- Household ID – Since no households were involved with the transit plans, the default value zero was used.
- Vehicle ID – Same as in the Transit Driver Plan file.
- ID of Starting Location – Beginning parking lot id.
- Vehicle Type – Light rail transit = 8, and buses = 5.

The file was 180 KB in size and contained approximately 9400 records.

8. Process Link Table

The extent of the case study network prohibited the inclusion of all possible process links in the microsimulation. Instead, a subset of process links was used. All activities were connected to the nearest parking location, with emphasis placed on matching activities and parking lots located on the same link. Process links were also used to join transit and activities that share a common link. If an activity had been placed on a link with no transit stops, no process links were created. The process link data included:

- `ID`
- `FROMID`
- `FROMTYPE` – Activity, Parking, or Transit Stop.
- `TOID`
- `TOTYPE` – Activity, Parking, or Transit Stop.
- `DELAY` – Set to 23 seconds.
- `COST` – No cost was incurred for traveling on the process links.
- `NOTES` – Given the default entry `NONE`.

The process link table, which contains approximately 526,000 records, is 24.4 MB.

9. LANE CONNECTIVITY TABLE

The lane connectivity table was generated by the *LaneConnectivity* program. However, there were approximately two dozen areas in the network with specialized connectivity that needed to be altered by hand. When analyzing the traveler plan files, it was obvious that certain links, particularly freeways and ramps that were expected to have a large number of travelers, in fact, had very few or no vehicles utilizing them. This was most often due to the fact that the *LaneConnectivity* program cannot account for turn barriers. For example, highways are sometimes connected to a secondary street by more than one ramp and the vehicles can turn only from one direction onto a ramp due to turn restrictions. The generated lane connectivity allowed traffic to turn both ways from the secondary street onto the ramps. So, the router-generated paths tended to put the traffic more on one ramp than the other for some beneficial reason, such as the ramp being shorter than the other. Manually changing the lane connectivity table to more realistic probability solved this problem. Also, several transit routes required U-turns because of the way the paths were constructed. The needed U-turns were also added.

- `NODE`
- `INLINK`
- `INLANE`
- `OUTLINK`
- `OUTLANE`
- `NOTES` – Given the default entry `NONE`.

The lane connectivity table, which has approximately 507,700 records, is 12.7 MB.

10. TRAFFIC CONTROLS

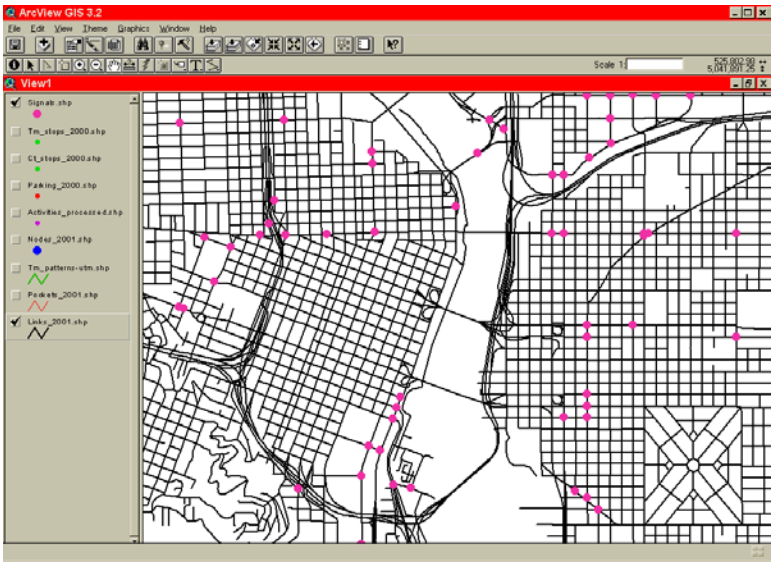

Fig. 11. Signalized nodes in the Portland network.

The traffic control tables (Unsignalized Node, Signalized Node (Fig. 11), Phasing Plan, Timing Plan, Signal Coordinator, and Detector tables) were generated automatically using the *CreateTrafficControls* program (see TRANSIMS Ver. 2.1 Volume Two (*Networks and Vehicles*), Section 1.6.3 (*CreateTrafficControls*) for more information).

The default values for the following network configuration keys are as follows:

`NET_ACTUATED_ALGORITHM_B_BETA`	1 meters / second
`NET_ACTUATED_ALGORITHM_B_DENSITY_CONST`	0 meters
`NET_ACTUATED_ALGORITHM_B_FLOW_CONST`	0.1 / seconds
`NET_DETECTOR_PRESENCE_SAMPLE_TIME`	1 second

These values were obtained from the study of actuated signal behavior (see the general actuated signal behavior documentation). In order to take advantage of the concept that well-chosen green times are necessary for coordinated signal operation, a signal-generation algorithm was developed where the minimum green length for a phase is based on a weighted sum of the number of incoming lanes participating in the phases, but it is never less than 25 seconds. (A permanent lane has a weight of 1, and a pocket lane has a weight of 1/3.) The total cycle length of at least 60 seconds is apportioned to the phases with constant yellow and red clearance intervals, and minimum green intervals based on the ratio of the number of incoming lanes in this phase to the total number of lanes. The green extension fraction is set to 60%, but the green extension is never permitted to be less than 12 seconds. Volume Seven (*Appendix: Scripts, Configuration*

Files, Special Travel Time Functions),Chapter Eleven (*MS-7*) lists the complete set of TRANSIMS configuration settings used to generate the signals.

Several modifications to the automatically generated traffic controls were made. Any yield signs that led from on-ramps onto freeways, as well as signals that were generated on freeway links, were removed. Intersections that encompassed multiple nodes in the network caused by offset links were combined into one node or the traffic controls were removed from most or all of the nodes to allow traffic to flow more realistically. Also, U-turns at signals were permitted where the connectivity allowed them.

Approximately 0.25% of the nodes were not accounted for in either the Signalized Node or Unsignalized Node tables. These nodes were generally one-way links (such as a freeway link) located on the edges of the network with traffic flowing in toward Portland. Because of this, none of the nodes had an "incoming" link, so they could not be assigned a record in the Unsignalized Node table. The link would also not require a signal with traffic flowing in only one direction.

The files created by the program varied in size from 19 KB (Signal Coordinator table) to 4.7 MB (Unsignalized Node table).

11. TRANSIMS TABLES NOT USED IN CASE STUDY

Several of the tables were not utilized in the case study:

- Speed Table
- Lane Use Table
- Barrier Table
- Turn Prohibition